Fast and lightweight on-device TTS with Tacotron2 and LPCNet

Vadim Popov, Stanislav Kamenev, Mikhail Kudinov, Sergey Repyevsky, Tasnima Sadekova, Vitalii Bushaev, Vladimir Kryzhanovskiy, Denis Parkhomenko

Huawei Technologies Co. Ltd., Russia

vadim.popov@huawei.com, stanislav.kamenev@huawei.com

Abstract

We present a fast and lightweight on-device text-to-speech system based on state-of-art methods of feature and speech generation i.e. Tacotron2 and LPCNet. We show that modification of the basic pipeline combined with hardware-specific optimizations and extensive usage of parallelization enables running TTS service even on low-end devices with faster than real-time waveform generation. Moreover, the system preserves high quality of speech without noticeable degradation of Mean Opinion Score compared to the non-optimized baseline. While the system is mostly oriented on low-to-mid range hardware we believe that it can also be used in any CPU-based environment.

Index Terms: on-device speech synthesis, recurrent neural vocoders, TTS optimization

1. Introduction

Neural models have recently become a common solution for Text-to-Speech synthesis since they can produce high-quality speech at reasonable computational costs. State-of-the-art results have been achieved by using attention-based models such as Tacotron [1] for feature generation and autoregressive WaveNet vocoder [2] for conditional waveform generation.

As long as Tacotron operated on speech frames and also was capable of predicting several speech frames in one step [1, 3], the speech generation module was considered as the main computational bottleneck. The improvements of the original autoregressive WaveNet vocoder were made in the three directions: 1) development of a non-autoregressive vocoder providing the same quality of speech [4, 5, 6]; 2) finding a lightweight architecture of neural network for conditional speech generation [7, 8]; 3) decreasing number of prediction steps for the given target sample rate [7, 9, 10].

While several solutions referenced above [7, 8, 9, 10] were claimed to be capable of faster-than-realtime on-device speech generation the requirements to the hardware are still high and can be met only on higher-end devices which are still too expensive for many people.

Curiously enough, although relatively complex schemes of parallel waveform generation for recurrent and autoregressive models have been proposed in [7] the simplest brute-force methods have not been extensively discussed in the literature, at least to our knowledge. For example, the input mel-spectrogram can be split into independent parts based on frame energy criterion so each part can be processed by a separate copy of the vocoder. Then the synthesized waves can be concatenated with or without post-processing techniques (e.g. cross-fading [11]). Such an approach though being used be practitioners (e.g. [12, 13]) is not normally referenced in literature even as a baseline.

In this paper, we combine various techniques of TTS model optimization and parallelization which can be either hardware-dependent or independent. Our solution is based on modified versions of two autoregressive neural architectures, namely Tacotron2 [3] and LPCNet [8], and is capable of generating high-quality speech. We show that this model is also efficient in terms of speed and memory usage. An extensive human evaluation conducted for four languages reveals that the overall quality of the synthesized speech is high enough and the optimization tricks we use do not lead to sound quality degradation.

The paper structure is as follows: in Section 2 we describe design of the feature generation module based on Tacotron2; in Section 3 various vocoder optimization techniques are discussed; in Section 4 human evaluation results are presented along with efficiency measurement; we conclude in Section 5.

2. Feature generation module

As mentioned above, our feature generation module is based on Tacotron2. Tacotron2 is a sequence-to-sequence neural architecture with attention [14] carrying out a transformation from an input character sequence into an output mel-spectrogram. Tacotron2 makes use of an autoregressive decoder implemented as a 2-layer LSTM with location-sensitive attention [15] and convolutional postnet module which was found critical for generating crisp mel-spectrograms. The integration of Tacotron2 and LPCNet suggested replacing output mel-spectrum features [16] of the original Tacotron2 with the native features of LPCNet i.e. 20-dimensional vector consisting of 18 Bark-scale cepstral coefficients (BFCC) [17] and 2 pitch parameters (period, correlation). We also found that predicting normalized BFCCs made training of the feature generation model more stable so we chose the option of predicting normalized features and adding a final denormalization layer.

The feature generation module and the vocoder can be effectively parallelized due to the big difference in their execution times. The overall execution time thus becomes almost equal to that of LPCNet. To facilitate this we fixed two computational bottlenecks of the original architecture.

The first bottleneck was created by a high computational cost of Tacotron2 decoder comprising a 2-layer LSTM with 1024 units. We chose to decrease the size of the LSTM by a factor of 4. We should note though that for some languages we had to increase the number of LSTM layers from 2 to 3 to prevent quality degradation.

The second bottleneck was caused by a wide receptive field of the postnet module. As far as the convolutions in the postnet are not causal we have to wait until the number of frames processed by the decoder matches the size of the receptive field. It directly influences the *first frame delay* or *FFD* i.e. a time delay between getting the character input and starting outputting the sound. For a zero-padded sequence and a stack of 1d-convolutions *FFD* is calculated as:

$$FFD = E + D * \lceil R/2 \rceil + P + V, \qquad (1)$$

where E is the encoder execution time, V is the vocoder per sample execution time, R is the width of the receptive field of the convolution stack, D is the decoder per frame execution time and P is the postnet per frame execution time.

The original postnet consists of 5 convolutions of shape 5×1 with stride 1, so $R = 21$. It implies a delay of $E+11D+P$ msec before the wave generation starts. To alleviate this problem we change widths of the convolution kernels to $[5, 3, 3, 3]$ thus reducing the receptive field to 11 and decreasing the wave generation delay to $E + 6D + P$. The encoder execution time E, in turn, is almost negligible compared to $6D$ so there is no much use in decreasing it. However, we should note that for the vanilla LPCNet Equation 1 is not completely true because of the *frame-rate subnet* [8] so in our final design, we integrated postnet of Tacotron2 and the frame-rate subnet of LPCNet into a single module. The resulting frame-rate module carried out the following function: 1) postnet transformation of Tacotron2; 2) BFCC denormalization and 3) frame-rate feature generation of LPCNet.

Finally, to improve synthesis of long sentences we replace location-sensitive attention with dynamic convolutional attention [18].

3. Vocoder parallelization

Our investigations were carried out for LPCNet [8] – an autoregressive vocoder based on recurrent neural networks but in principle the methods described in this section should be applicable (probably with slight modifications) to other recurrent vocoders, e.g. to WaveRNN [7].

In general, the main disadvantage of autoregressive vocoders suitable for mobile devices is that they can't enjoy the benefits of parallel computations: their autoregressive nature makes independent parallel generation difficult. Moreover, such models are usually composed of small compact layers, so using parallel computations for matrix operations also does not make sense because the overhead on creating and synchronizing threads in this case is too high compared to the speedup gain due to parallelization.

However, sometimes it is possible to synthesize some parts of the speech signal in a parallel manner independently so that a simple concatenation of the synthesized waves produces a good record with no artifacts at the borders of the parts. We refer to the frames that serve as such borders as *splitting frames*.

3.1. Splitting frames detection

If two adjacent words in a sentence are separated with a distinct pause, the waveforms corresponding to these words can be considered approximately independent, so synthesizing these two words in parallel should not lead to a loss in sound quality. Likewise, since speech samples that correspond to unvoiced sounds are almost uncorrelated, parallel synthesis of speech parts separated with unvoiced frames is also possible. This simple intuition is the basis of the energy-based criterion of splitting frame detection.

LPCNet inputs are BFCCs [17], so applying inverse discrete cosine transform to these coefficients results in an approximation of speech signal log-energy in the neighbourhood of certain frequencies located uniformly on Bark scale. The frames containing silence can be characterized by low overall energy whereas frames corresponding to unvoiced sounds have most of their energy located at high frequencies (see Figure 1). Thus, frames at which a spectrogram can be divided into nearly independent parts can be detected with the following energy-based criterion: either (1) overall energy in a frame is less than a threshold B_{sil}, or (2) ratio of energy at high frequencies to energy at low frequencies is greater than a threshold B_{unv}.

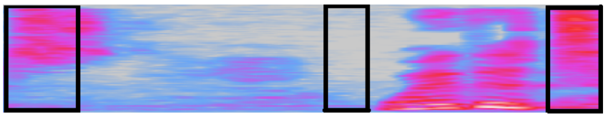

Figure 1: *Boxes on the spectrogram correspond to silence or unvoiced sounds.*

Though being quite simple and intuitive, the energy-based approach to finding splitting frames requires manual tuning of the thresholds B_{sil} and B_{unv}. That is why we decided to try another approach and to solve splitting frames detection task with neural networks. We chose a compact architecture similar to LPCNet encoder: two $1D$ convolution layers with kernel size 3, output channels number 32 and $tanh$ activations are followed by a fully-connected layer with 32 units and $tanh$ activation and the last sigmoid layer. So, the inputs were composed of frame-level acoustic features (BFCCs and 16-dimensional pitch embedding) and the output was a single number i.e. the probability that splitting spectrogram at the current frame did not lead to audible artifacts.

Below we reformulate the requirement not to introduce audible artifacts in terms of the loss function. We start from an observation that splitting spectrogram at "wrong" frames always results in one particular type of defects i.e. in a clicking sound on that frame and in an apparent vertical stripe at the corresponding segment of the spectrogram (see Figure 2). So, we can detect clicks by comparing two spectrograms corresponding to speech signal synthesized with and without parallelization. We chose the following function for the measure of the perceived click strength:

$$\mu(S_{par}, S_{std}) = \left(\max_{i\in\mathcal{L}\cap\mathcal{I}} \frac{\log^2 E_{par}^{(i)}}{\log E_{std}^{(i)}} + \max_{i\in\mathcal{H}\cap\mathcal{I}} \frac{\log E_{par}^{(i)}}{\log E_{std}^{(i)}} \right)^2 \tag{2}$$

where we denote spectrum at the analyzed frame by S, energy at a certain frequency i by $E^{(i)}$, the set of low frequencies by $\mathcal{L}$, the set of high frequencies by $\mathcal{H}$, the set of frequencies i for which $\log E_{par}^{(i)} > \log E_{std}^{(i)} > 1$ by $\mathcal{I}$ and subscripts par and std stand for synthesis with and without parallelization correspondingly.

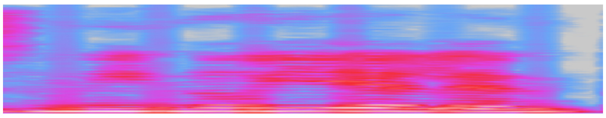

Figure 2: *Splitting frames corresponding to vowels produce clicks (vertical stripes on the spectrogram).*

There are a few points we need to make about Formula 2: 1) we divide frequency range into two sets $\mathcal{L}$ and $\mathcal{H}$ because we found that in some cases energy of the clicks is concentrated either in the high or low part of the spectrum; 2) our approximation of log-energy obtained from BFCCs isn't always accurate, so we choose maximum to serve as robust aggregation function; 3) we square the numerator of the first term to put higher penalty for low-frequency noise and decrease perceived strength of the click; 4) we exclude all cases when the standard version produces higher energy sound so we consider only ratios exceeding one; if no term satisfies this condition, the maximum is set to zero.

The resulting loss function penalizes frames splitting at which results in clicks and, on the other hand, encourages splitting at frames when no click is observed:

$$Loss(x) = p_\theta(x) \cdot \mu(x) + (1 - p_\theta(x)) \cdot C \tag{3}$$

where x denotes the analyzed frame, $p_\theta(x)$ is the probability (given by the neural network with parameters θ) that this frame can be used to divide spectrogram into parts synthesized independently without loss in quality, $\mu(x)$ is the measure of a click (calculated by Equation 2) that appears if this frame is considered as splitting one and C is some positive borderline value: if the measure of a click $\mu(x)$ is less than C, we regard it as "no click".

To train such a network we create a dataset consisting of pairs of audio records synthesized with and without parallelization. To generate audio with parallelization for this dataset we allocate splitting frames at random. After the network is trained, it is also possible to create a new dataset where splitting frames are allocated in accordance with $p_\theta(x)$ rather than randomly and to continue the training on the new dataset. However, we found that such a training procedure that resembles imitation learning [19] does not improve quality.

At test time, frames with $p_\theta(x) > 0.95$ were considered for splitting. Neural models found about 10% more splitting frames than the energy-based algorithm for all languages we tested.

3.2. Cross-fading

Instead of detecting frames that split spectrogram into independent parts so that the corresponding synthesized waveforms can simply be concatenated without any post-processing, we can focus on improving the post-processing techniques that work well for any choice of a splitting frame. This can be done with decent quality if the segments synthesized in parallel are overlapped.

As a baseline we take *linear cross-fading* technique [11]. Assume that the vocoder is processing two segments in parallel so that the first segment contains frames $1, .., k$ while the second one contains frames $k, k+1, ..$ That is, two waves overlap in k-th frame. Concatenating the first wave $s^{(1)}$ and the second wave $s^{(2)}$ with linear cross-fading means that in k-th frame samples of the resulting wave s are given by:

$$s_i = (1 - a_i)s_i^{(1)} + a_i s_i^{(2)}, \quad i = 0, .., N \tag{4}$$

where N is the number of samples in a frame. In our experiments we took $a_i = i/N$ so that $s^{(1)}$ fades out uniformly.

An ideal choice of coefficients in the linear combination (4) is known to depend on correlation between $s^{(1)}$ and $s^{(2)}$ (see [11]). Intuitively, it is clear that for highly correlated signals the quality of concatenation with cross-fading is less dependent on the values of coefficients in Equation 4. In the limiting case when $s^{(1)}$ is equal to $s^{(2)}$ the values of these coefficients do not even matter once they sum to one. It suggests that cross-fading quality can improve after applying a left shift to the second signal $s^{(2)}$ such that its correlation with $s^{(1)}$ increases. This idea is illustrated in Figure 3.

Thus, we consider *linear cross-fading with shift*:

$$m = \operatorname*{arg\,min}_{j=0,..,M} \sum_{i=0}^{W} |s_{i+j}^{(2)} - s_i^{(1)}| \tag{5}$$

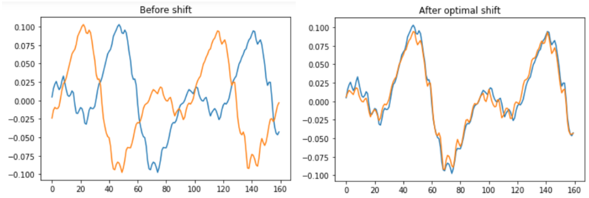

Figure 3: *Unmodified waves (left) and the same waves but with shift applied to one of them (right). Linear cross-fading works better for the two waves on the right.*

Table 1: *A/B testing results.*

Which is better?	Non-parallel	Parallel	Identical
EB-splitting	23.9%	21.5%	54.6%
NN-splitting	21.1%	19.4%	59.5%
XF with shift	14.4%	17.5%	68.1%
Which is better?	W/o shift	With shift	Identical
Cross-fading	7.3%	43.3%	49.4%

$$s_i = \left(1 - \left(\frac{i}{N}\right)^\alpha\right) s_i^{(1)} + \left(\frac{i}{N}\right)^\alpha s_{i+m}^{(2)}, \quad i = 0, .., N \tag{6}$$

where M is the maximum possible shift value and W is size of the window used for calculation $L1$ distance between signals (we found that minimizing $L1$ distance leads to the same quality as maximizing correlation). We put $M = W = N/2$ in our experiments. We also introduce additional parameter α: the larger α is, the longer the first wave $s^{(1)}$ does not fade out (high cross-fading quality was observed for $\alpha \in [1; 3]$).

The code for splitting frames detection (both energy-based and network-based methods) and synthesis with linear cross-fading (both with and without shift) is available at https://github.com/li1jkdaw/LPCNet_parallel/tree/code.

4. Performance evaluation

We performed subjective human evaluation tests on Amazon Mechanical Turk. Four single-speaker datasets were used: we trained models on male Italian and female English, French and Spanish speakers. All the datasets are internal except the English one which is LJSpeech dataset [20]. A small portion of audio records that were used in our experiments is available at https://li1jkdaw.github.io/LPCNet_parallel.

4.1. A/B testing

In order to check that the methods of vocoder parallelization described in Section 3 do perform well we conducted a series of A/B tests. In each of these tests the participants were presented with 25 pairs of recordings of the same sentence and asked to choose which one they preferred in case they heard any difference. In each pair the records were synthesized conditioned on the same spectrogram since we did not want a small variation in the Tacotron2 output to affect the result. The main purpose of the tests was to ensure that the parallelization did not lead to degradation of the sound quality. As we had several approaches to vocoder parallelization we carried out separate A/B tests for different system design choices. For the strategy involving synthesis of non-overlapping segments we tested two splitting frame detection methods i.e. energy-based and neural

Table 2: *Mean Opinion Scores for ground truth records and speech synthesized with different methods.*

Dataset	Duration	Vocoder	Ground Truth	Non-parallel	EB-splitting	NN-splitting	XF with shift
English	24 hours	WaveRNN	4.46 ± 0.17	4.08 ± 0.20	4.15 ± 0.22	—	**4.22 ± 0.20**
English	24 hours	LPCNet	3.98 ± 0.12	3.74 ± 0.10	**3.78 ± 0.10**	3.71 ± 0.11	3.75 ± 0.11
Italian	23 hours	LPCNet	4.15 ± 0.14	3.45 ± 0.16	**3.60 ± 0.16**	3.42 ± 0.26	3.56 ± 0.19
French	8 hours	LPCNet	4.46 ± 0.10	3.83 ± 0.17	3.86 ± 0.16	**3.88 ± 0.17**	3.84 ± 0.19
Spanish	17 hours	LPCNet	4.43 ± 0.12	**3.54 ± 0.11**	3.52 ± 0.11	3.50 ± 0.12	3.50 ± 0.18

network-based criteria (EB-splitting and NN-splitting respectively). For the alternative strategy involving synthesis of overlapping segments we tested the cross-fading with shift as the post-processing technique (XF with shift). In the latter case, we allocated 2 splitting frames per second. Each pair in all tests was evaluated by at least 20 listeners. The tests were performed on English data only.

The results of these tests are presented in Table 1. In more than half of cases people noticed no difference between parallel and non-parallel versions. In the remaining cases the difference between the presented methods was small. To obtain statistically significant results, we applied sign test [21] and concluded that we can't reject (at 95% confidence level) the hypothesis that speech synthesized with the analyzed methods of parallelization has the same quality as the one synthesized in a normal way.

We also performed another A/B test to show that the linear cross-fading with shift leads to better sound quality than the same method without shift (XF w/o shift). Sign test applied to the results of Table 1 allows us to reject the hypothesis that the cross-fading without shift works at least as well as the one with shift. When a splitting frame corresponds to a vowel, linear cross-fading without shift leads to audible artifacts. In contrast to the previous A/B test with overlapping segments, for this one we allocated 10 splitting frames per second instead of 2 to underline the mentioned problem and check if adding shift can fix it.

4.2. MOS evaluation

To evaluate the overall quality and naturalness of the speech synthesized with our TTS system, we launched Mean Opinion Score (MOS) evaluation for four languages. Additionally, we trained Tacotron2 + WaveRNN system on English dataset sampled at 22kHz to show that the vocoder optimization methods described in Section 3 can be applied not only to LPCNet or 16kHz synthesis. For each test the TTS system produced no less than 15 audio records.

We asked participants selected according to a geographic criterion to estimate quality of these records on five-point Likert scale, i.e. to classify a record as “Bad” (1 point), “Poor” (2 points), “Fair” (3 points), “Good” (4 points) or “Excellent” (5 points). We also included ground truth recordings and special noisy recordings in the tests in order to keep track of the attention of the assessors and prevent random answers. The assessors who gave less than 3 points to the ground truth records or more than 3 points to the noisy records were excluded from the experiment. As in A/B tests, each piece of audio was evaluated by at least 20 people.

Table 2 demonstrates that all optimization tricks referenced there perform well enough in general and show the same sound quality as TTS with non-parallel vocoder in particular. Note that in case of WaveRNN we used 22kHz audio which explains the difference between ground truth MOS in English tests with LPCNet and WaveRNN.

Table 3: *Vocoder parallelization efficiency.*

	1 thread		2 threads		3 threads	
	FFD	RTF	FFD	RTF	FFD	RTF
MT6762	323	1.64	345	1.07	352	**0.89**
Kirin950	202	1.18	213	**0.75**	243	**0.67**
Kirin710	170	1.09	184	**0.69**	191	**0.56**

4.3. Efficiency evaluation

To test overall performance we implemented Tacotron2 and LPCNet (with splitting frames detection by energy-based criterion) on several mobile devices with 8 core ARM processors: Mediatek MT6762 (4x2.0GHz Cortex-A53 + 4x1.5GHz Cortex-A53), Kirin950 (4x2.3GHz Cortex-A72 + 4x1.8GHz Cortex-A53) and Kirin710 (4x2.2GHz Cortex-A73 + 4x1.7GHz Cortex-A53).

The whole TTS application requires only 12.5Mb of storage (11.4Mb for Tacotron2 and 1.1Mb for LPCNet). All weights are stored as 8-bit numbers. As for the speed, we report average values of Real Time Factor (RTF) and First Frame Delay (FFD, see Section 2) in Table 3. RTF is defined as the time it takes to synthesize some piece of an audio divided by its duration. FFD is measured in milliseconds.

Table 3 shows that using 3 threads for parallel vocoder gives almost 2x speedup resulting in faster than real-time synthesis. At the same time, independent generation of non-overlapping segments introduces an overhead of approximately 5 msec on the detection of splitting frames. Moreover, we should note that as the splitting time is undefined the RTF can vary from phrase to phrase.

5. Conclusion

In this work we have presented a text-to-speech system that is suitable for low-to-mid range mobile devices. Optimization techniques that we describe allow this system to run on low-end hardware without any loss in quality. Besides, we investigated several parallelization techniques applicable to autoregressive vocoders and showed that these techniques did not have any negative impact on the synthesized speech despite the fact that parallelization breaks the correlation between speech samples. Further research can be focused on the development of lightweight TTS vocoders capable of generating any speech segment without splitting frame detection or segment overlap.

6. References

[1] Y. Wang, R. Skerry-Ryan, D. Stanton *et al.*, "Tacotron: Towards end-to-end speech synthesis," *ArXiv*, 2017. [Online]. Available: https://arxiv.org/abs/1703.10135

[2] A. van den Oord, S. Dieleman, H. Zen *et al.*, "WaveNet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 125–125.

[3] J. Shen, R. Pang, R. J. Weiss *et al.*, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*. IEEE, April 2018, pp. 4779–4783.

[4] A. van den Oord, Y. Li, I. Babuschkin *et al.*, "Parallel WaveNet: Fast high-fidelity speech synthesis," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 2018, pp. 3918–3926.

[5] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," *ArXiv*, 2018. [Online]. Available: http://arxiv.org/abs/1807.07281

[6] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019*. IEEE, May 2019, pp. 3617–3621.

[7] N. Kalchbrenner, E. Elsen, K. Simonyan *et al.*, "Efficient neural audio synthesis," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 10–15 Jul 2018, pp. 2410–2419.

[8] J.-M. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019*. IEEE, May 2019, pp. 5891–5895.

[9] Z. Jin, A. Finkelstein, G. J. Mysore, and J. Lu, "FFTNet: A real-time speaker-dependent neural vocoder," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*. IEEE, 2018, pp. 2251–2255.

[10] V. Popov, M. Kudinov, and T. Sadekova, "Gaussian LPCNet for multisample speech synthesis," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020*. IEEE, 2020.

[11] M. Fink, M. Holters, and U. Zölzer, "Signal-matched power-complementary cross-fading and dry-wet mixing," in *Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16)*, September 2016, pp. 109–112.

[12] [Online]. Available: https://github.com/fatchord/WaveRNN/issues/9

[13] [Online]. Available: https://ai.googleblog.com/2020/04/improving-audio-quality-in-duo-with.html

[14] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[15] J. Chorowski, D. Bahdanau, D. Serdyuk *et al.*, "Attention-based models for speech recognition," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA: MIT Press, 2015, p. 577–585.

[16] S. B. Davis and P. Mermelstein, "Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[17] B. Moore, *An introduction to the psychology of hearing*, 5th ed. Brill, 2012.

[18] E. Battenberg, R. J. Skerry-Ryan, S. Mariooryad *et al.*, "Location-relative attention mechanisms for robust long-form speech synthesis," *ArXiv*, vol. abs/1910.10288, 2019.

[19] T. Osa, J. Pajarinen, G. Neumann *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, p. 1–179, 2018.

[20] K. Ito, "The LJ Speech Dataset," 2017.

[21] W. Conover, *Practical nonparametric statistics*, 3rd ed. New York, NY [u.a.]: Wiley, 1999.